F23-068-D-NeuraSight

Project Team

Ahmed Iqbal	i20-0447
Mizrab Sheikh	i20-0453
Hissam Savul	i20-0780

Session 2023-2024

Supervised by

Dr. Muhammad Arshad Islam

Department of Computer Science

National University of Computer and Emerging Sciences
Islamabad, Pakistan

June, 2024

Contents

List of Figures

List of Tables

Abstract

NeuraSight addresses the pressing issue of financial fraud detection within transactional data through an innovative application. Users upload datasets and select from three detection options: Graph Neural Networks, Temporal Motifs, and Time Series Analysis. Graph Neural Networks create models from labeled or semi-labeled data for real-time fraud detection. Temporal Motifs offer customizable pattern analysis, allowing users to identify anomalies within the network. Time Series Analysis employs multi-variate techniques to uncover fraudulent trends. NeuraSight's comprehensive approach yields promising results, with each method contributing unique insights into fraudulent activities. The application enables users to visualize financial networks and execute intuitive queries, enhancing fraud detection accuracy and efficiency. NeuraSight stands as a versatile tool for proactive fraud prevention, providing users with tailored detection methods and empowering them to safeguard against financial threats.

Chapter 1

Introduction

In today's increasingly interconnected and data-driven financial landscape, the rapid rise in digital transactions has given rise to a corresponding surge in financial fraud. Detecting and preventing fraudulent activities within these intricate financial networks is a pressing concern for businesses and individuals alike. This project, aptly named NeuraSight, emerges as a pivotal response to this escalating challenge.

The foundation of this undertaking rests on the fundamental premise that the ability to discern anomalies and potential fraudulent activities within financial transactions is paramount. NeuraSight combines cutting-edge technologies, such as Graph Neural Networks, Temporal Motifs, and Time Series Analysis, to equip users with the tools they need to safeguard their financial operations.

This report serves as a comprehensive guide to NeuraSight, offering a clear understanding of the problem at hand and the motivations behind our work. As we delve further into these pages, we will elucidate how NeuraSight enables users to analyze and detect financial fraud in transactional data, empower themselves with data-driven insights, and contribute to a more secure financial landscape. The knowledge shared here not only outlines the purpose of our project but also highlights the valuable contributions it brings to the field of financial fraud detection, setting a new standard for proactive security in the digital age. **?**.

1.1 Problem Statement

Financial fraud poses a significant threat to organizations and individuals, leading to substantial monetary losses and reputational damage. Detecting fraudulent activities within transactional data is a complex challenge that demands advanced computational techniques. The existing solutions are often inadequate, requiring a comprehensive, user-friendly, and efficient application to empower financial managers and analysts in identifying fraudulent patterns.

The problem with existing fraud detection methods lies in their limited adaptability and robustness, primarily stemming from their reliance on static rules that prescribe specific criteria for fraud identification. These rigid criteria quickly grow obsolete as fraudsters adapt their tactics, compromising the system's effectiveness in detecting new and increasingly sophisticated fraud patterns. Adding to this challenge, traditional approaches typically restrict their analysis to a mere 2-4 hops within the network (cannot detect complex patterns), which, in actuality, is not enough for uncovering intricate and evolving fraud patterns. Additionally, these conventional techniques tend to yield many false positives, potentially flagging legitimate transactions as potential fraud, further hampering their utility.

This is where NeuraSight comes into play, providing versatile, cutting-edge techniques for transactional fraud detection, financial network visualization, and sophisticated queries to enhance fraud detection efforts.

1.2 Scope

The project is aimed at creating a user-friendly desktop application for detecting financial frauds. The goal is to help financial managers easily identify fraudulent activities in their transaction data. It's important to note that the primary objective of the application is fraud detection, rather than fraud prevention, although users can utilize the insights to enhance their systems' security.

The project will leverage advanced technologies like Graph Neural Networks, Temporal Motif Analysis, and Time Series Analysis to achieve this. These tools will provide financial managers with effective means of detecting fraud. The application will feature a straightforward interface, enabling users to upload data, choose from three different analysis methods, and visualize the results to support informed decision-making.

We will be putting efforts to prioritize optimizing performance, ensuring scalability, and offering comprehensive user support through user-friendly documentation. The aim is to develop Nuerasight as an exceptionally efficient and user-centric tool for identifying fraud in financial transactions.

1.3 Modules

Following are the proposed modules we used for our implementation of NeuraSight

1.3.1 Data Overview

NeuraSight offers users a comprehensive overview of their transactional datasets, providing insights into network structures, temporal patterns, and key statistical metrics. This holistic view empowers users to gain a deeper understanding of their financial data, facilitating informed decision-making and targeted fraud detection strategies.

1. Network Structures: Visualizes transactional statistics.
2. Temporal Patterns: Analyzes recurring temporal trends in data.
3. Statistical Metrics: Provides essential data distribution insights.

1.3.2 Time Series Analysis

Time Series Analysis: Leveraging state-of-the-art time series forecasting techniques, NeuraSight enables users to predict future trends and anomalies in financial data, providing proactive measures against potential fraudulent activities and enhancing decision-making capabilities.

1. Trend Prediction: Forecasts future trends.
2. Decision Support: Offers actionable insights for fraud prevention.

1.3.3 Temporal Motif Analysis

Motif Analysis: NeuraSight's motif analysis module offers users the flexibility to explore complex transactional networks, allowing them to uncover subtle anomalies and irregularities by customizing motifs or leveraging predefined patterns, enhancing fraud detection accuracy and efficiency.

1. Static Motifs: Identifies fixed patterns indicative of fraud.
2. Temporal Motifs: Uncovers evolving patterns over time.
3. Visualization: Provides visual representation of motif occurrences.
4. Custom Motif Queries: Allows users to define specific motif search criteria for targeted analysis.

1.4 Graph Neural Network (GNN)

Graph Neural Networks (GNN): Through sophisticated machine learning algorithms, NeuraSight's GNN module facilitates the creation of robust fraud detection models from labeled or semi-labeled transactional data, empowering users to continuously adapt and optimize their detection strategies in real-time, bolstering financial security.

1. Model Creation: Builds robust fraud detection models.
2. Real-time Adaptation: Continuously optimizes detection strategies.
3. Enhanced Security: Proactively identifies and prevents fraud.

User class	Description
Financial Managers	These are the primary users of the application. They are responsible for managing financial data and detecting fraudulent activities within their organization's transactional data. They need an efficient tool to streamline fraud detection and make data-driven decisions.
Data Analyst	Data analysts may assist financial managers in using the application and interpreting the results. They are responsible for understanding the technical aspects of the fraud detection techniques and optimizing the tool's usage.
Software Developers	The development team is responsible for creating and maintaining the application. They need to ensure that the software is functional, secure, and scalable. They will also implement and maintain the machine learning models and algorithms.
IT Administrators	IT administrators are responsible for the application's deployment, maintenance, and integration with the organization's IT infrastructure. They need to ensure the application runs smoothly and securely.

Example: User Classes and Characteristics

Chapter 2

Project Requirements

This chapter describes the functional and non-functional requirements of the project.

2.1 Use-case

The selection of the requirement gathering technique(s) will depend on the type of project. For instance,

- Use case (use case diagram + detail use case) is an effective technique for interactive end-user applications.

2.1.1 High Level Use-Cases

Use case diagrams

Use Case ID	UC - 01
Use Case Name	Import Transaction Data
Actors	Financial Manager
Type	Primary
Description	The system allows financial managers to import the data, view the relevant data and set the required columns.

Use Case ID	UC - 02
Use Case Name	Generate Data Overview
Actors	Financial Manager
Type	Primary

Description	The system allows financial managers to instantly generate an informative overview of their uploaded data, providing essential insights about data source, size, and variables.

Use Case ID	UC - 03
Use Case Name	Run Smart Fraud Detection
Actors	Financial Manager
Type	Primary
Description	The system employs Graph Neural Networks (GNN) to automatically detect fraudulent patterns and connections in the financial transaction data.

Use Case ID	UC - 04
Use Case Name	Network-Pattern Detection
Actors	Financial Manager
Type	Primary
Description	The system enables financial managers to uncover time-sensitive fraud patterns by selecting pre-defined temporal motifs.

Use Case ID	UC - 05
Use Case Name	Create Custom Network Patterns
Actors	Financial Manager
Type	Primary
Description	The system equips financial managers to craft custom temporal motifs effortlessly using an intuitive graphical interface or a flexible command-line tool.

Use Case ID	UC - 06
Use Case Name	Run Time Series Analysis
Actors	Financial Manager
Type	Primary
Description	The system equips financial managers with advanced tools for time series analysis accessible through a user-friendly GUI or command-line interface.

Use Case ID	UC - 07
Use Case Name	Visualize Analyzed Network
Actors	Financial Manager
Type	Primary
Description	The system provides a dynamic visualization window to present results, making it effortless for users to interpret detected motifs, time series data, and other critical insights.

2.1.2 Expanded Use-Cases

Use case diagrams

<table>
<tr><td>Use Case ID</td><td>UC - 01</td></tr>
<tr><td>Use Case Name</td><td>Import Transaction Data</td></tr>
<tr><td>Actors</td><td>Financial Manager</td></tr>
<tr><td>Stakeholders and Interests</td><td>Financial Manager (Upload data)
System (Imports data)</td></tr>
<tr><td>Pre-conditions</td><td>The user has a dataset that they want to import.
The dataset is in CSV format.</td></tr>
<tr><td>Post-conditions</td><td>Transactional data is successfully imported</td></tr>
<tr><td>Main Success Scenario</td><td><table>
<tr><th>User</th><th>System</th></tr>
<tr><td>1. User clicks on the "Import Data" button.</td><td></td></tr>
<tr><td></td><td>2. The system provides an interface for data upload.</td></tr>
<tr><td>3. User selects data dataset for upload.</td><td></td></tr>
<tr><td></td><td>4. The system processes, uploads and validates the data.</td></tr>
<tr><td>5. Successfully uploaded data is available for further use.</td><td></td></tr>
</table></td></tr>
<tr><td>Alternatives</td><td><table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1a. User cancels data upload</td><td></td></tr>
<tr><td>4a. The data is not in the required format so the system highlights the format error and prompts for data reupload.</td><td></td></tr>
</table></td></tr>
<tr><td>Technology and Data Variations List</td><td>Data format variations
Data source variations
Upload method variations</td></tr>
<tr><td>Frequency of Occurrence</td><td>Frequent</td></tr>
</table>

Table 2.8: Use Case: Import Transaction Data

<table>
<tr><td>Use Case ID</td><td>UC - 02</td></tr>
<tr><td>Use Case Name</td><td>Generate Data Overview</td></tr>
<tr><td>Actors</td><td>Financial Manager</td></tr>
<tr><td>Stakeholders and Interests</td><td>Financial Manager (Understands data insights)
System (Provides data overview)</td></tr>
<tr><td>Pre-conditions</td><td>Data is uploaded and available.</td></tr>
<tr><td>Post-conditions</td><td>Data overview is displayed.</td></tr>
<tr><td>Main Success Scenario</td><td>
<table>
<tr><td>User</td><td>System</td></tr>
<tr><td>1. User selects “Generate Data Overview.”</td><td></td></tr>
<tr><td></td><td>2. The system generates a comprehensive data overview tailored to financial data.</td></tr>
<tr><td></td><td>3. The system performs data analysis, including:
Total transaction count.
Total transaction volume.
Average transaction amount.
Transaction frequency and distribution.
Summary statistics of numerical data.
Categorization of transaction types.</td></tr>
<tr><td>4. User reviews the overview.</td><td></td></tr>
</table>
</td></tr>
<tr><td>Alternatives</td><td>None</td></tr>
<tr><td>Technology and Data Variations List</td><td>None</td></tr>
<tr><td>Frequency of Occurrence</td><td>Frequent</td></tr>
</table>

Table 2.9: Generate Data Overview Use Case

<table>
<tr><td>Use Case ID</td><td>UC - 03</td></tr>
<tr><td>Use Case Name</td><td>Run Smart Fraud Detection</td></tr>
<tr><td>Actors</td><td>Financial Manager</td></tr>
<tr><td>Stakeholders and Interests</td><td>Financial Manager (Detect fraud patterns)
System (Detects fraud patterns)</td></tr>
<tr><td>Pre-conditions</td><td>Requisite data is uploaded and accessible within the system.</td></tr>
<tr><td>Post-conditions</td><td>Detection and categorization of potential fraud patterns with detailed report provision.</td></tr>
<tr><td>Main Success Scenario</td><td>
<table>
<tr><td>User</td><td>System</td></tr>
<tr><td>1. User selects "Run Smart Fraud Detection."</td><td></td></tr>
<tr><td></td><td>2. The system prompts the user to choose a Graph Neural Network Model.</td></tr>
<tr><td>3. The user selects the required Graph Neural Network Model</td><td></td></tr>
<tr><td></td><td>3. The system initiates the Graph Neural Network (GNN) model for fraud detection.</td></tr>
<tr><td></td><td>4. The GNN model analyzes financial data to detect fraudulent patterns and connections.</td></tr>
<tr><td></td><td>5. Detected fraud patterns are categorized and the user is provided with information on potential fraud instances, such as transaction details and confidence scores.</td></tr>
<tr><td>6. User reviews the fraud detection report.</td><td></td></tr>
</table>
</td></tr>
<tr><td>Alternatives</td><td>5a. No Fraud patterns were found
6a. User exits.</td></tr>
<tr><td>Technology and Data Variations List</td><td>GNN model
Variations in fraud patterns</td></tr>
</table>

<table>
<tr><td>Use Case ID</td><td>UC - 04</td></tr>
<tr><td>Use Case Name</td><td>Network-Pattern Detection</td></tr>
<tr><td>Actors</td><td>Financial Manager</td></tr>
<tr><td>Stakeholders and Interests</td><td>Financial Manager (Identify time-sensitive fraud patterns)
System (Analyzes data)</td></tr>
<tr><td>Pre-conditions</td><td>Requisite data is uploaded and accessible within the system.</td></tr>
<tr><td>Post-conditions</td><td>Detection and categorization of potential fraud patterns with detailed report provision.</td></tr>
<tr><td>Main Success Scenario</td><td><table>
<tr><th>User</th><th>System</th></tr>
<tr><td>1. User selects "Network-Pattern Detection".</td><td></td></tr>
<tr><td></td><td>2. System provides options for selecting pre-defined temporal motifs.</td></tr>
<tr><td>3. User selects a motif/s for analysis.</td><td></td></tr>
<tr><td></td><td>4. The system validates the user's motif selection.</td></tr>
<tr><td></td><td>5. System analyzes data using the chosen motif/s.</td></tr>
<tr><td>6. User reviews the analysis results.</td><td></td></tr>
</table></td></tr>
<tr><td>Alternatives</td><td>2a. User creates custom motifs
5a. No Fraud patterns were found
6a. User exits.</td></tr>
<tr><td>Technology and Data Variations List</td><td>Variations in motif selection and analysis techniques</td></tr>
<tr><td>Frequency of Occurrence</td><td>Occasional</td></tr>
</table>

Table 2.11: Network-Pattern Detection Use Case

<table>
<tr><td>Use Case ID</td><td>UC - 05</td></tr>
<tr><td>Use Case Name</td><td>Create Custom Network Patterns</td></tr>
<tr><td>Actors</td><td>Financial Manager</td></tr>
<tr><td>Stakeholders and Interests</td><td>Financial Manager (Custom motif creation)
System (Enables motif creation)</td></tr>
<tr><td>Pre-conditions</td><td>Requisite data is uploaded and accessible within the system.</td></tr>
<tr><td>Post-conditions</td><td>Detection and categorization of potential fraud patterns with detailed report provision.
Custom motifs are created.</td></tr>
<tr><td>Main Success Scenario</td><td><table>
<tr><th>User</th><th>System</th></tr>
<tr><td>1. User selects "Create Custom Network Patterns."</td><td></td></tr>
<tr><td></td><td>2. System provides a user-friendly GUI for creating custom temporal motifs.</td></tr>
<tr><td>3. User designs a custom motif.</td><td></td></tr>
<tr><td></td><td>4. The system validates the custom motif to ensure it adheres to predefined criteria or constraints.</td></tr>
<tr><td>5. The system asks the user to confirm the saving of the motif.</td><td></td></tr>
<tr><td>6. The user acknowledges the confirmation.</td><td></td></tr>
<tr><td></td><td>7. The system saves the custom motif to the so that it is available for usage.</td></tr>
</table></td></tr>
<tr><td>Alternatives</td><td><table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>4a. The created motif is invalid.</td><td>Repeat steps 1 - 3.</td></tr>
</table></td></tr>
<tr><td>Technology and Data Variations List</td><td>None</td></tr>
<tr><td>Frequency of Occurrence</td><td>Occasional</td></tr>
</table>

Table 2.12: Use Case: Create Custom Network Patterns

<table>
<tr><td>Use Case ID</td><td>UC - 06</td></tr>
<tr><td>Use Case Name</td><td>Run Time Series Analysis</td></tr>
<tr><td>Actors</td><td>Financial Manager</td></tr>
<tr><td>Stakeholders and Interests</td><td>Financial Manager (Analyze time series data)
System (Supports analysis)</td></tr>
<tr><td>Pre-conditions</td><td>Requisite data is uploaded and accessible within the system.</td></tr>
<tr><td>Post-conditions</td><td>Detection and categorization of potential fraud patterns with detailed report provision.</td></tr>
<tr><td>Main Success Scenario</td><td><table>
<tr><th>User</th><th>System</th></tr>
<tr><td>1. User selects "Run Time Series Analysis."</td><td></td></tr>
<tr><td></td><td>2. The system provides options for time series analysis and choosing relevant data variables to show as plots.</td></tr>
<tr><td>3. The user configures the analysis settings based on their specific analytical goals.</td><td></td></tr>
<tr><td></td><td>4. The system performs the time series analysis using the chosen configuration.</td></tr>
<tr><td>5. Results are displayed to the user, presenting insights and trends in the time series data.</td><td></td></tr>
</table></td></tr>
<tr><td>Alternatives</td><td><table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>3a. The user chooses different variables</td><td></td></tr>
</table></td></tr>
<tr><td>Technology and Data Variations List</td><td>Variations in analysis methods.
Variations in the selected data variables for analysis.</td></tr>
<tr><td>Frequency of Occurrence</td><td>Occasional</td></tr>
</table>

Table 2.13: Use Case: Run Time Series Analysis

<table>
<tr><td>Use Case ID</td><td>UC - 07</td></tr>
<tr><td>Use Case Name</td><td>Visualize Analyzed Network</td></tr>
<tr><td>Actors</td><td>Financial Manager</td></tr>
<tr><td>Stakeholders and Interests</td><td>Financial Manager (Interpret analysis results)
System (Provides visualization)</td></tr>
<tr><td>Pre-conditions</td><td>Requisite data is uploaded and accessible within the system.</td></tr>
<tr><td>Post-conditions</td><td>Detection and categorization of potential fraud patterns with detailed report provision.
Results are visualized.</td></tr>
<tr><td>Main Success Scenario</td><td><table>
<tr><th>User</th><th>System</th></tr>
<tr><td>1. User performs fraud detection using GNN's, Motif Analysis or Time Series Analysis.</td><td></td></tr>
<tr><td></td><td>2. The system opens a visualization window with options.</td></tr>
<tr><td>3. The user interacts with the visualization window to: - View detected motifs. - Examine time series data. - Explore other insights.</td><td></td></tr>
<tr><td></td><td>4. The system offers interactive visualizations, such as: - Graphs for motifs and connections. - Time series plots and trends. - Heatmaps and statistical visualizations.</td></tr>
</table></td></tr>
<tr><td>Alternatives</td><td><table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1a. The user wants to change the configurations for fraud detection methods</td><td></td></tr>
<tr><td>4a. The system was unresponsive</td><td></td></tr>
</table></td></tr>
<tr><td>Technology and Data Variations List</td><td>Visualization technology variations.</td></tr>
<tr><td>Frequency of Occurrence</td><td>Frequent</td></tr>
</table>

Table 2.14: Use Case: Visualize Analyzed Network

2.2 Functional Requirements

This section describes the functional requirements of the system expressed in the natural language style. This section is typically organized by feature as a system feature name and specific functional requirements associated with this feature. It is just one possible way to arrange them. Other organizational options include arranging functional requirements by use case, process flow, mode of operation, user class, stimulus, and response depend on what kind of technique has been used to understand functional requirements. Hierarchical combinations of these elements are also possible, such as use cases within user classes.

2.2.1 Module 1 - Data Overview

Following are the functional requirements for the Data Overview module:

1. Data Presentation: The application shall present the uploaded transactional dataset in a visually appealing format, utilizing graphs, charts, and figures to provide a comprehensive overview of the data.

2. Customization: Users shall have the option to customize the presentation of data, including selecting specific variables, time periods, and aggregation methods to tailor the displayed information to their preferences.

3. Interactive Features: The data overview module shall include interactive features such as zooming, filtering, and sorting to enable users to explore the dataset efficiently.

4. Export Functionality: Users shall be able to export the displayed data in various formats, including CSV, Excel, and PDF, for further analysis or reporting purposes.

2.2.2 Module 2 - Time Series Analysis

Following are the functional requirements for the Time Series Analysis module:

1. Time Series Forecasting: Users shall have access to time series forecasting capabilities to predict future trends and behaviors based on historical transactional data.

2. Statistical Analysis: Users shall have access to a range of statistical tools and techniques, including trend analysis, seasonality detection, and correlation analysis, to gain insights into the temporal behavior of financial transactions.

3. Visualization: The time series analysis module shall provide interactive visualization tools, such as line charts, heatmaps, and histograms, to visualize time series data and analysis results effectively.

4. Prediction Capability: The application shall include predictive modeling capabilities to forecast future trends and potential fraudulent activities based on historical transactional data.

2.2.3 Module 3 - Motif Analysis

Following are the functional requirements for the Motif Detection module:

1. Motif Identification: The application shall automatically detect temporal motifs, or users can define custom motifs, within the transactional dataset to uncover recurring patterns indicative of fraudulent behavior.

2. Pattern Analysis: Users shall be able to analyze detected motifs to determine their significance and potential implications for fraud detection.

3. Graphical Representation: The motif detection module shall provide graphical representations of detected motifs, highlighting their occurrences and relationships within the transactional data network.

4. Interactive Exploration: Users shall have the ability to interactively explore detected motifs, zooming in/out and filtering the data to gain deeper insights into suspicious transaction patterns.

2.2.4 Module 4 - GNN Analysis

Following are the functional requirements for the GNN Analysis module:

1. Model Training: The application shall train graph neural network models using labeled or semi-labeled transactional data to detect patterns indicative of financial fraud.

2. Model Evaluation: Users shall be able to evaluate the performance of trained GNN models using metrics such as accuracy, precision, recall, and F1 score.

3. Real-time Prediction: The GNN analysis module shall enable users to apply trained models to real-time transactional data streams for immediate fraud detection and mitigation.

4. Model Interpretation: The application shall provide tools for interpreting GNN model predictions, including feature importance analysis and visualization of decision-making processes.

2.3 Non-Functional Requirements

This section specifies nonfunctional requirements. These quality requirements should be specific, quantitative, and verifiable. The following are some examples of documenting guidelines.

2.3.1 Reliability

Usability is crucial for NeuraSight to facilitate efficient fraud detection by users. The application should be intuitive, allowing users to navigate through features seamlessly. Error avoidance and recovery mechanisms must be integrated to minimize user mistakes and ensure a smooth experience. For example, NeuraSight should provide users with intuitive interfaces, clear instructions, and error prompts for incorrect inputs, enhancing overall usability.

2.3.2 Scalability

NeuraSight should be designed to scale seamlessly to accommodate increasing data volumes and user loads. The system must support horizontal and vertical scaling strategies to ensure optimal performance as user demand grows. Scalability requirements should address resource allocation, load balancing, and system architecture to maintain efficiency and responsiveness. For example, NeuraSight should dynamically allocate resources to handle peak loads during periods of high user activity, ensuring uninterrupted service and minimizing latency.

2.3.3 Usability

Usability requirements for NeuraSight are essential to facilitate user interaction and optimize fraud detection efforts. The application should be intuitive and easy to learn, allowing users to navigate through features effortlessly. Error avoidance and recovery mechanisms should be integrated to minimize mistakes and enhance the user experience. For instance, NeuraSight should provide users with the ability to retrieve previous transactions with a single interaction, reducing the time and effort required for data retrieval.

2.3.4 Performance

NeuraSight must meet specific performance benchmarks to ensure efficient processing of transactional data. Performance requirements should address various system operations, including graph neural network analysis, motif identification, and time series analysis. For example, NeuraSight should download and process transactional datasets within a specified time frame, even when dealing with large volumes of data, to maintain responsiveness and efficiency.

2.3.5 Security

Security is paramount for NeuraSight to protect sensitive financial data and prevent unauthorized access. Robust security measures, such as access control mechanisms, and compliance with industry standards, are necessary to safeguard the system and its data. For example, NeuraSight should implement authentication mechanisms to verify user identities and prevent unauthorized access. Regular security audits and updates should also be conducted to mitigate potential vulnerabilities and ensure ongoing protection.

2.3.6 Documentation

Comprehensive documentation is essential for NeuraSight to facilitate system understanding, maintenance, and troubleshooting. Documentation requirements should include user manuals, technical guides, API documentation, and release notes. The documentation must be clear, organized, and up-to-date, providing users with detailed instructions on system usage, configuration, and troubleshooting procedures. Additionally, NeuraSight should maintain version-controlled documentation to track changes and ensure accuracy over time.

2.4 Domain Model

Create a representation of the domain model for your project.

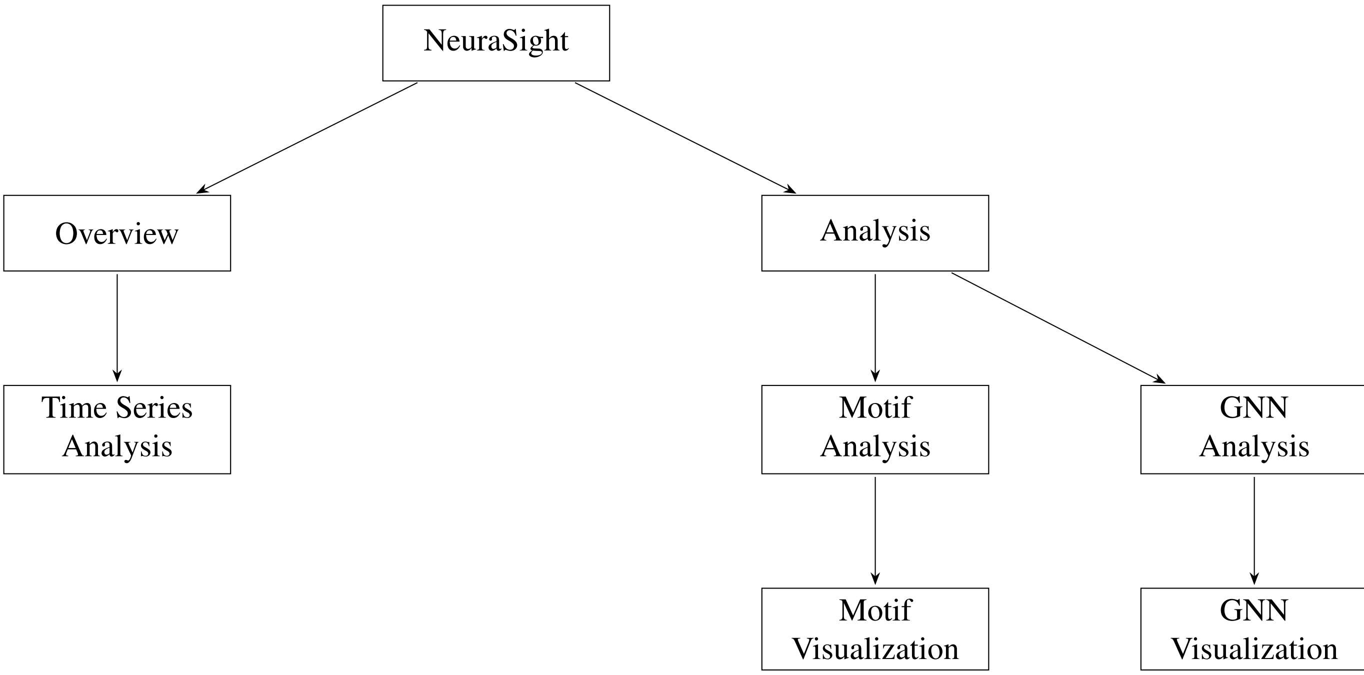


Figure 2.1: Simplified Domain Model Diagram for NeuraSight

Chapter 3

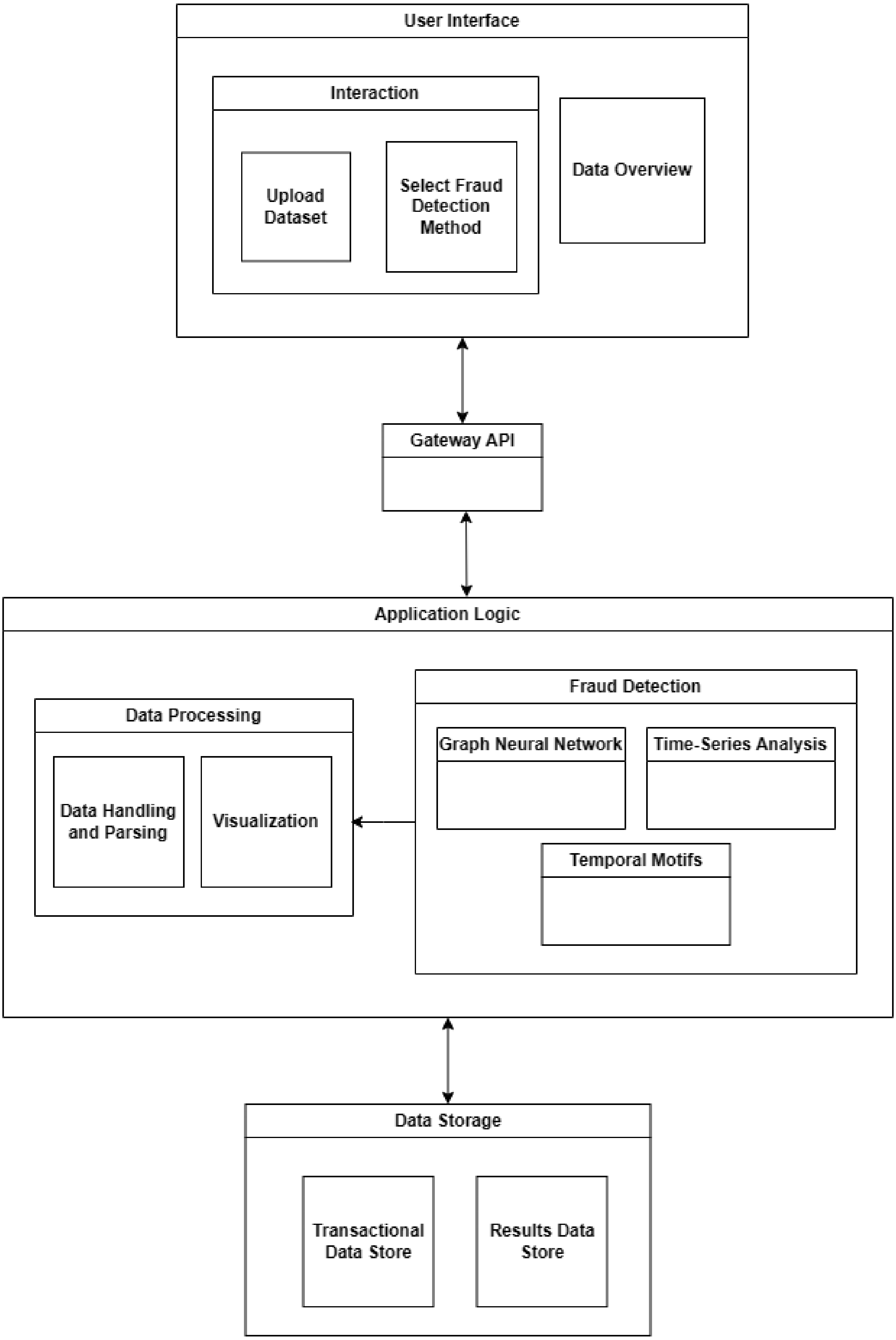
System Overview

3.1 Architectural Design

The user interface serves as the gateway for user interaction, facilitating effortless engagement with the system's functionalities. It simplifies the process of dataset uploads, commonly in CSV format, ensuring a continuous influx of fresh data for analysis. Additionally, users are empowered to customize their fraud detection approach by selecting from a range of methods such as "graph neural network" or "time-series analysis," allowing them to tailor the detection process to their specific requirements.

Embedded within the system's core, the application logic orchestrates the flow of data and governs crucial transformations. It meticulously handles data parsing and preparation, undertaking tasks like cleaning and feature extraction to ensure data readiness for fraud detection. Leveraging chosen methods like Graph Neural Networks and Time-Series Analysis, the application logic executes fraud detection, identifying intricate fraud patterns and anomalies within the transactional data.

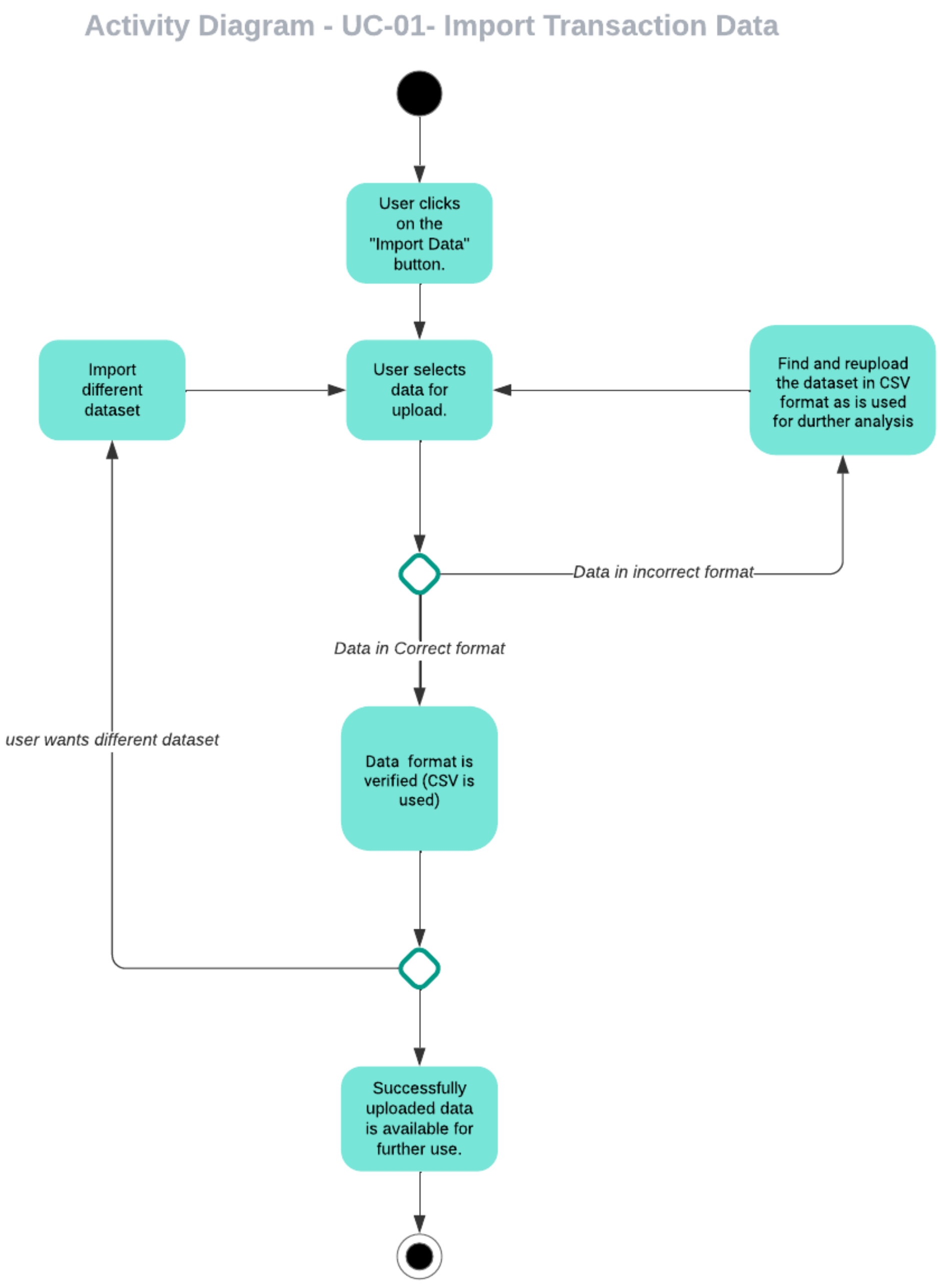
The data storage component serves as the backbone of the system, ensuring data availability, integrity, and accessibility. It houses the raw transactional data, preserving it for ongoing analysis and reference. Additionally, the component archives the outcomes of fraud detection processes, including flagged transactions and detected fraud types, providing valuable insights for analysis and decision-making.


User Interface
Interaction
Upload Dataset
Select Fraud Detection Method
Data Overview
Gateway API
Application Logic
Data Processing
Data Handling and Parsing
Visualization
Fraud Detection
Graph Neural Network
Time-Series Analysis
Temporal Motifs
Data Storage
Transactional Data Store
Results Data Store

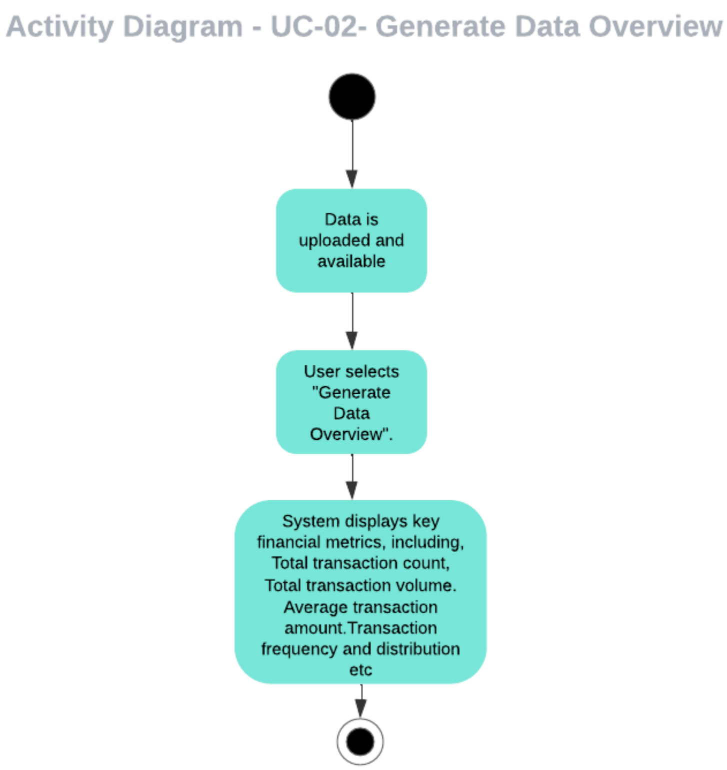
3.2 Design Models

3.2.1 Activity Diagram

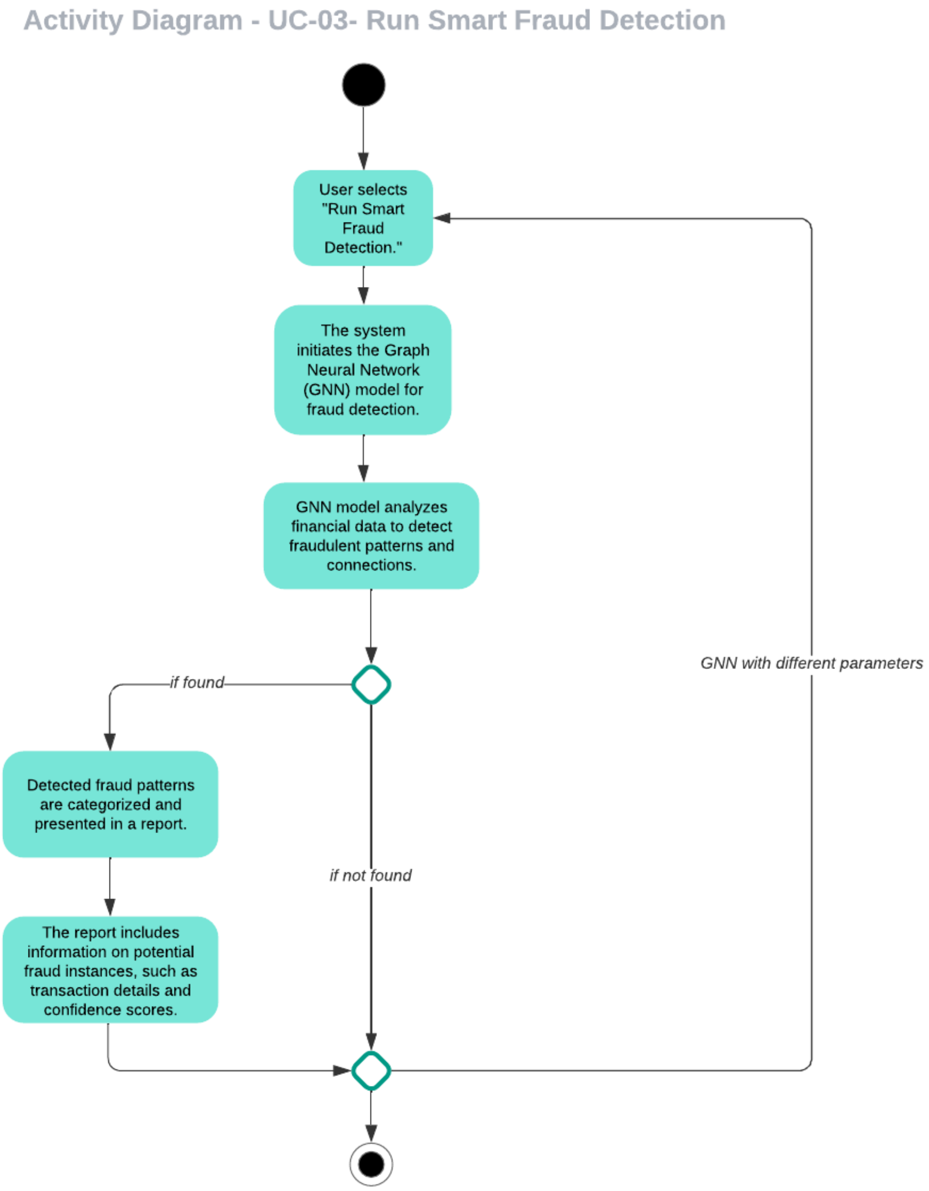
3.2.1.1 UC-01-Import Transaction Data



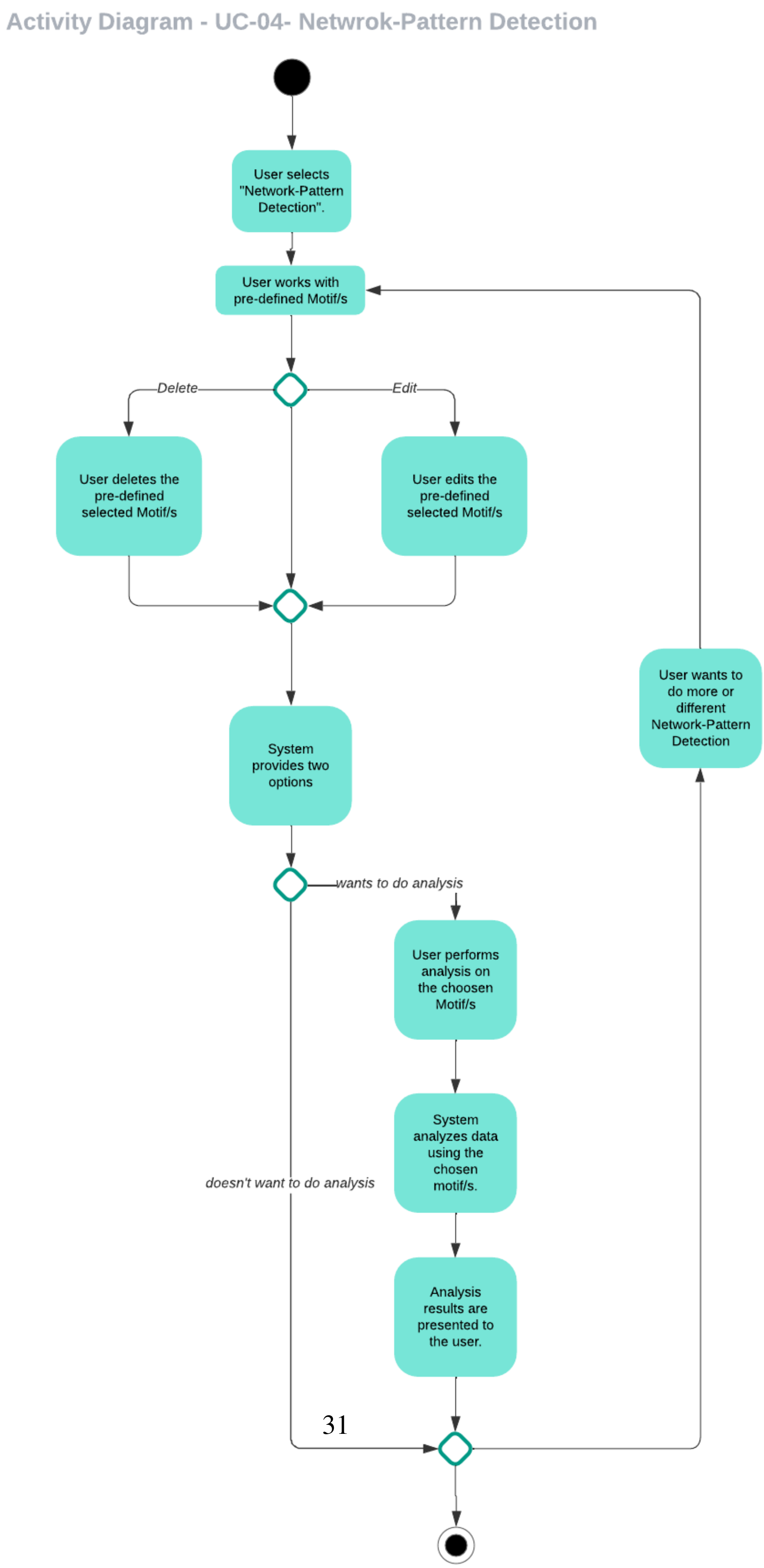
3.2.1.2 UC-02-Generate Data Overview



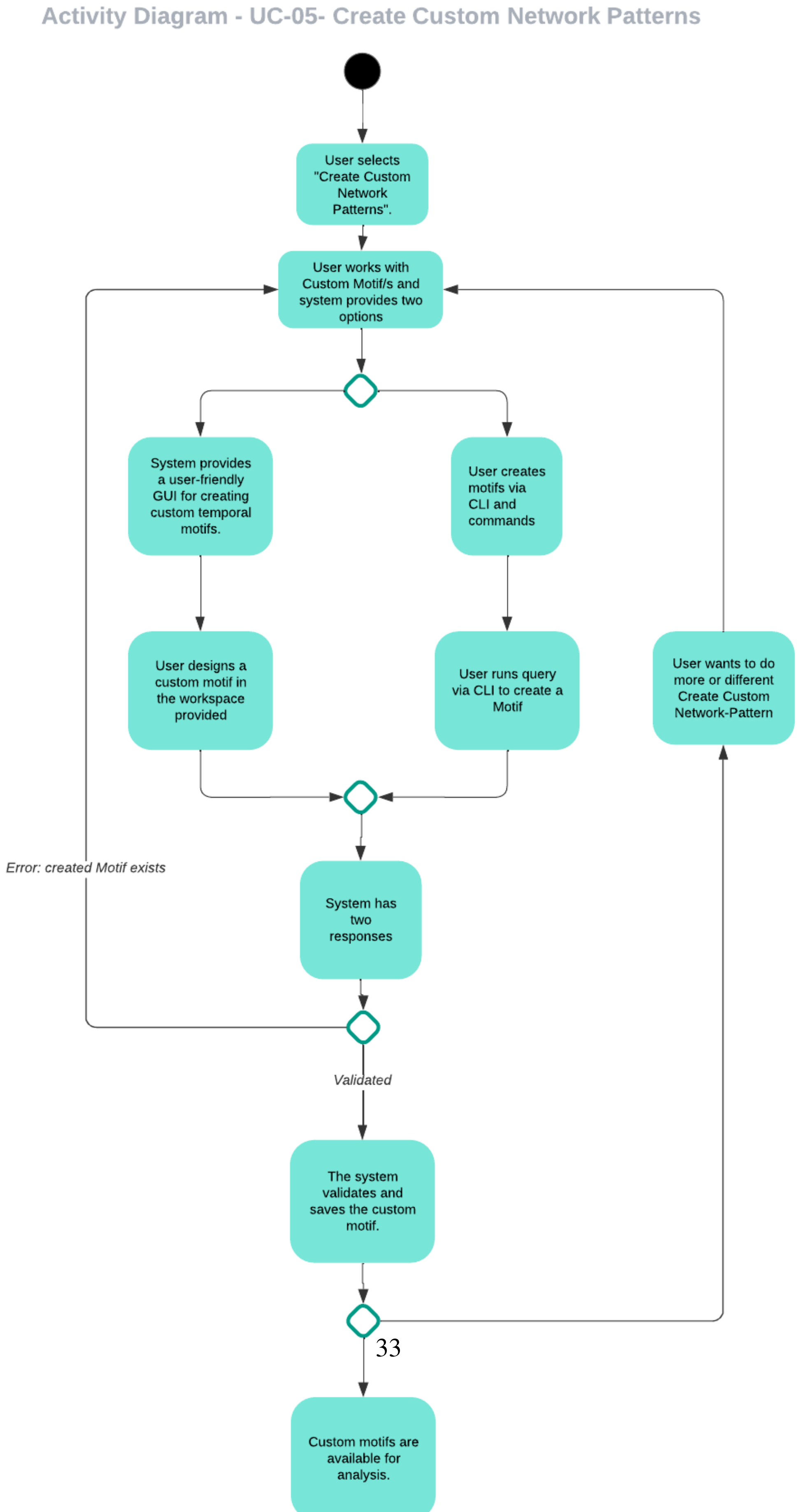
3.2.1.3 UC-03-Run Smart Fraud Detection



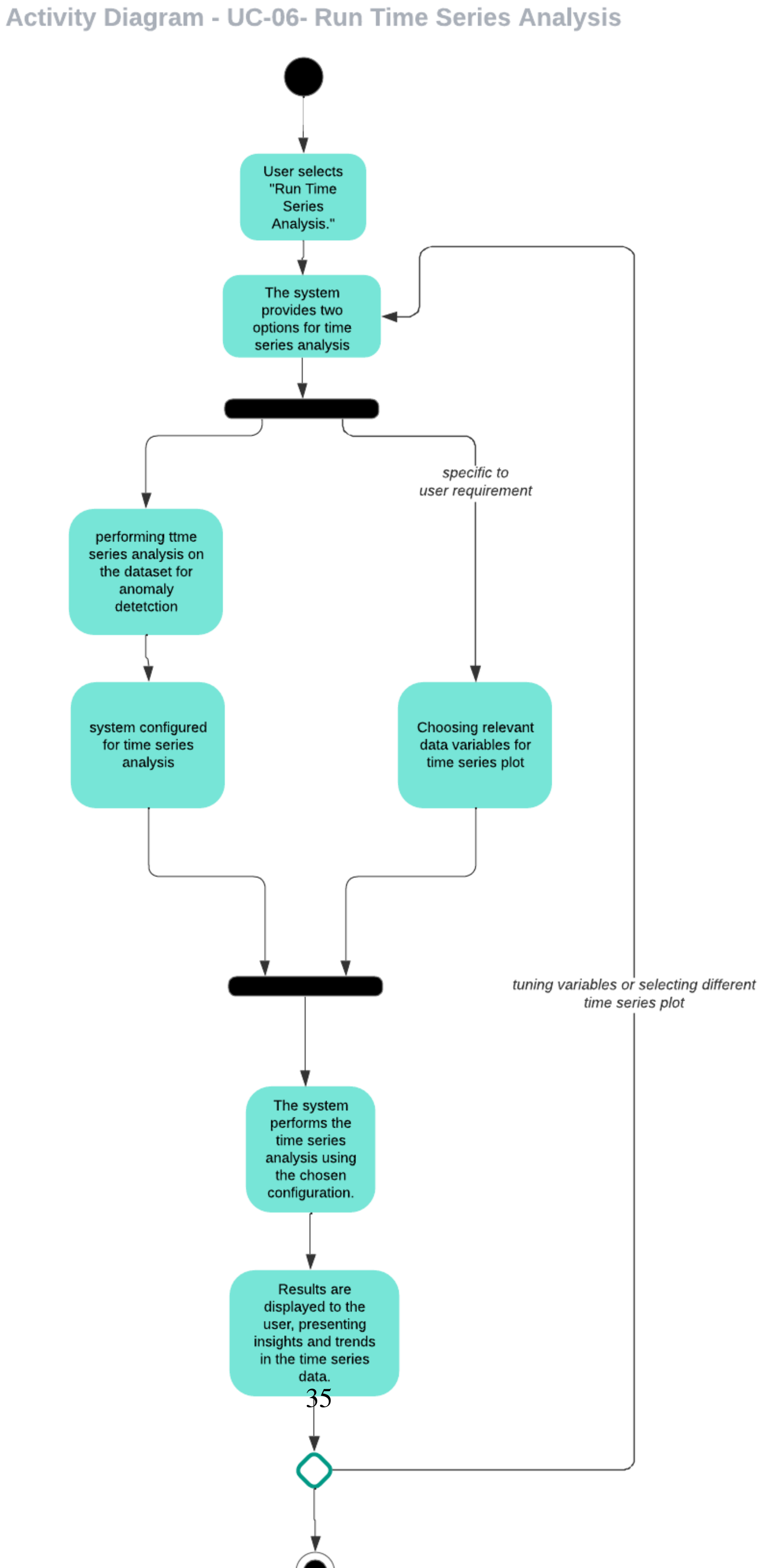
3.2.1.4 UC-04-Network-Pattern Detection



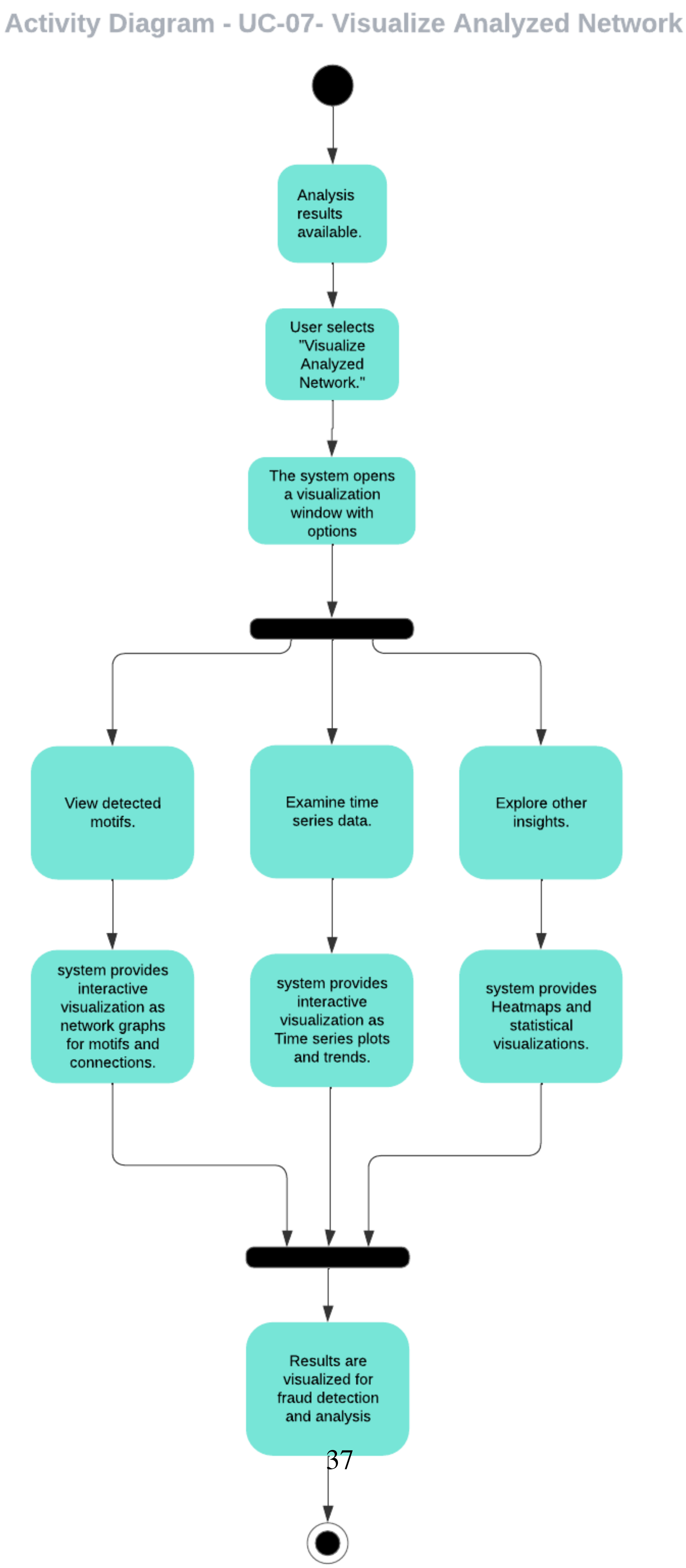
3.2.1.5 UC-05-Create Custom Network Patterns



3.2.1.6 UC-06-Run Time Series Analysis



3.2.1.7 UC-07-Visualize Analyzed Network



3.2.2 Data Flow Diagram

3.2.2.1 Level 0

3.2.2.2 Level 1

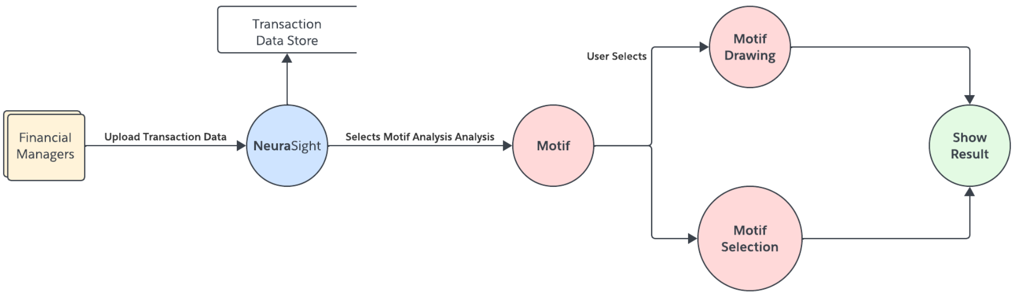


Figure 3.1: Temporal Motifs Module

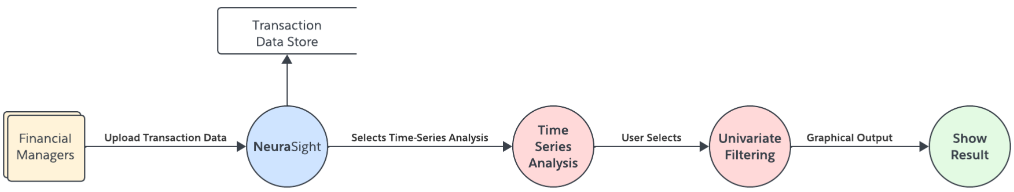


Figure 3.2: Time Series Analysis Module

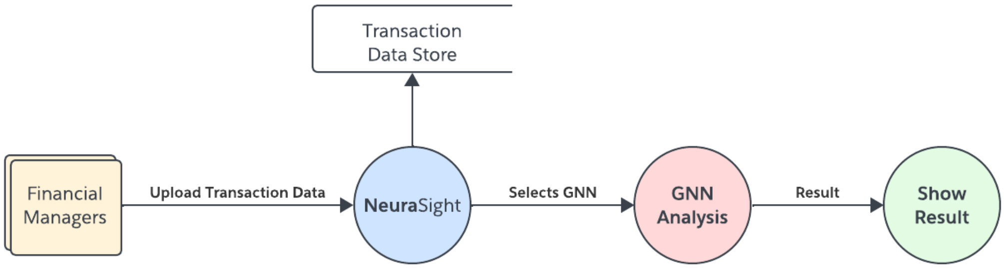


Figure 3.3: Graph Neural Networks Module

3.2.3 System Sequence Diagram

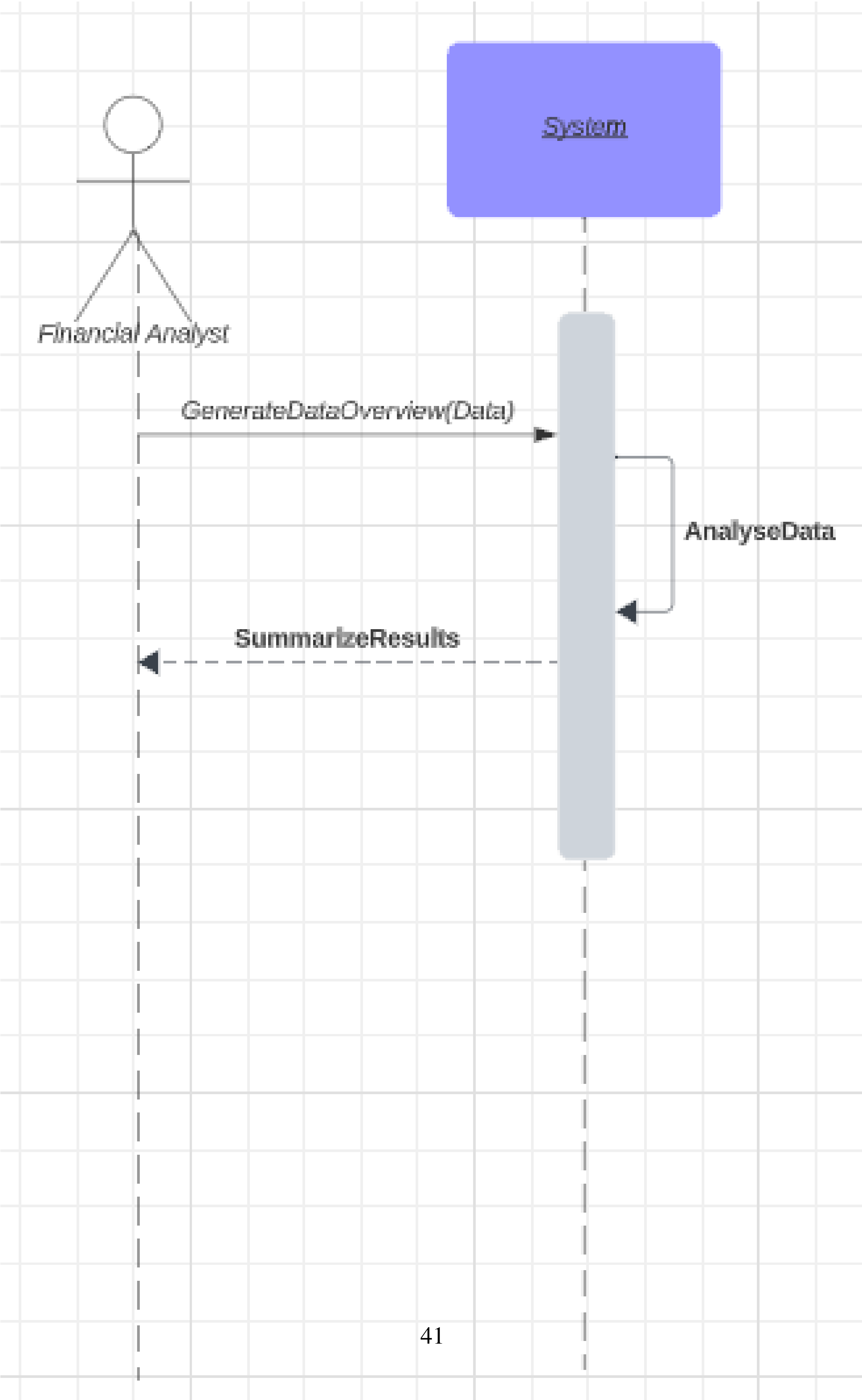


Figure 3.4: Data Overview

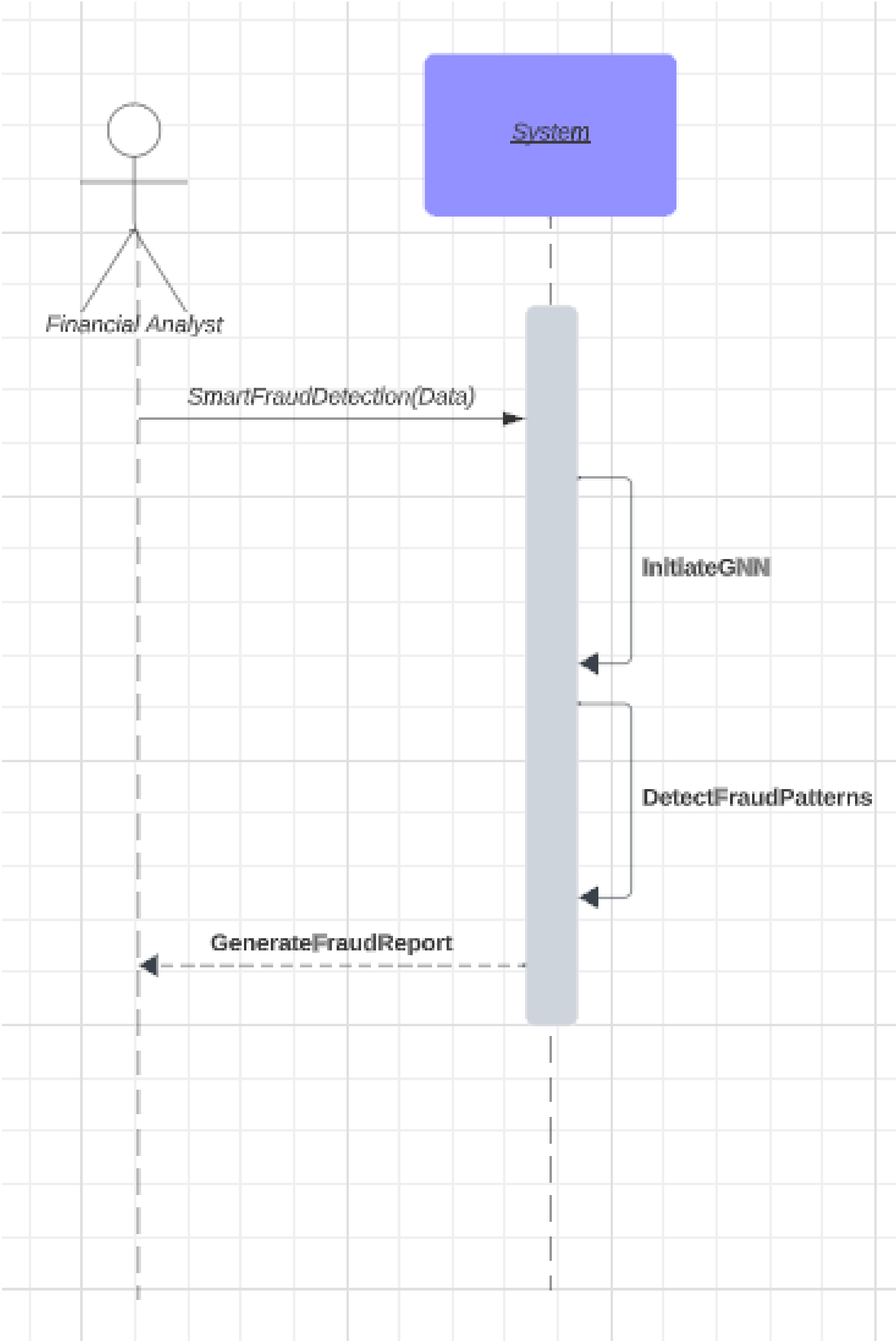


Figure 3.5: Graph Neural Networks

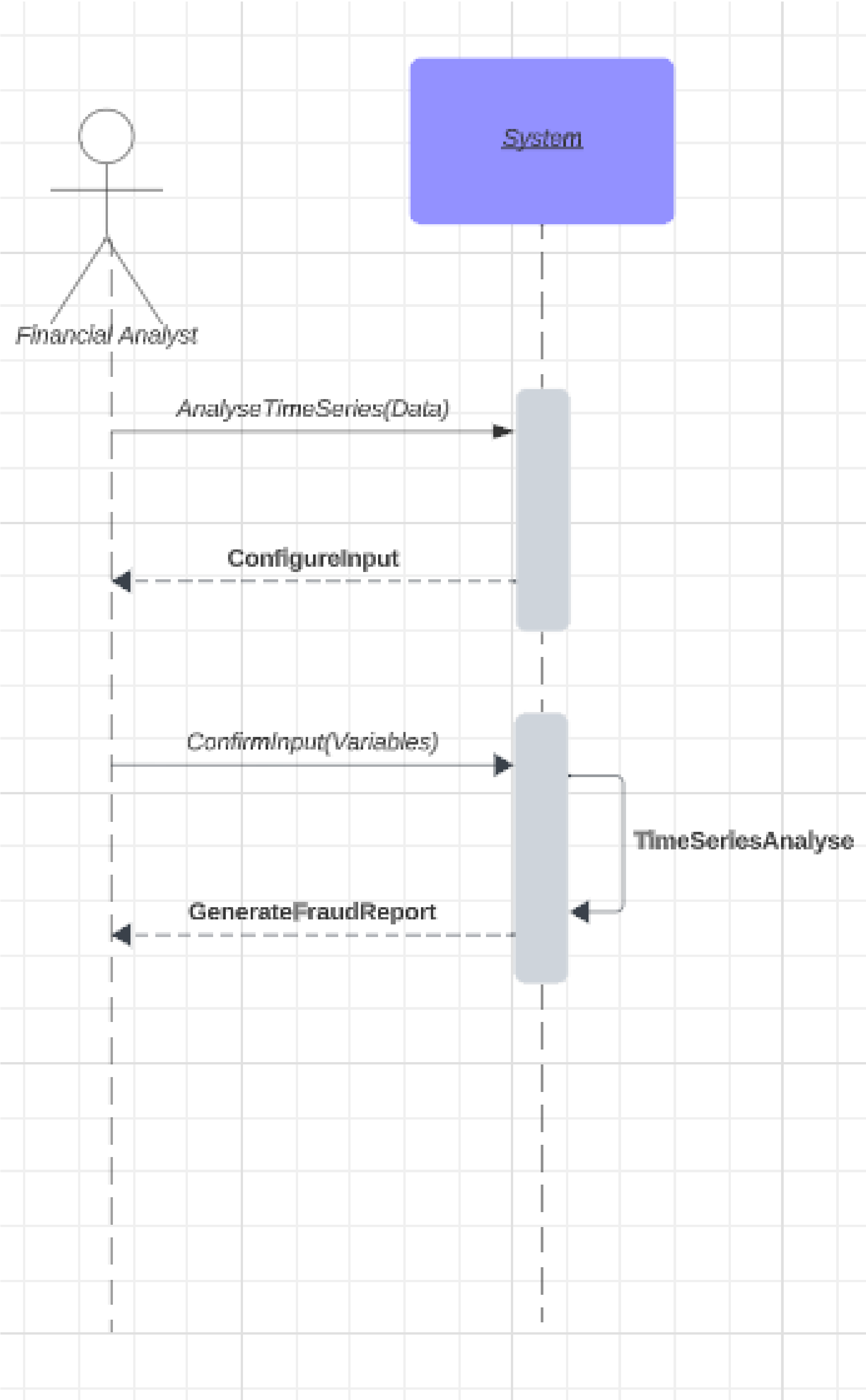
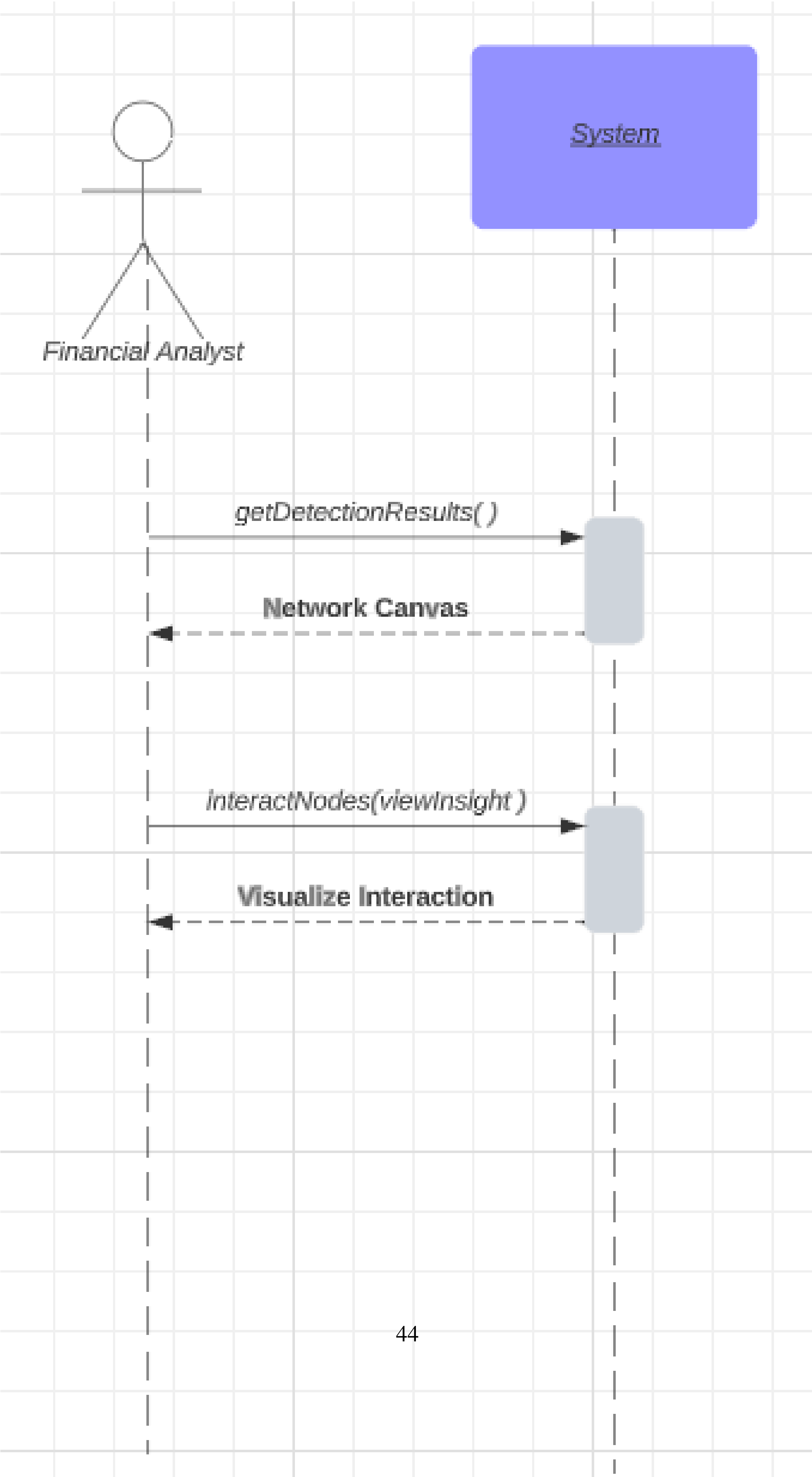


Figure 3.6: Time Series


System
Financial Analyst
getDetectionResults( )
Network Canvas
InteractNodes(viewInsight)
Visualize Interaction

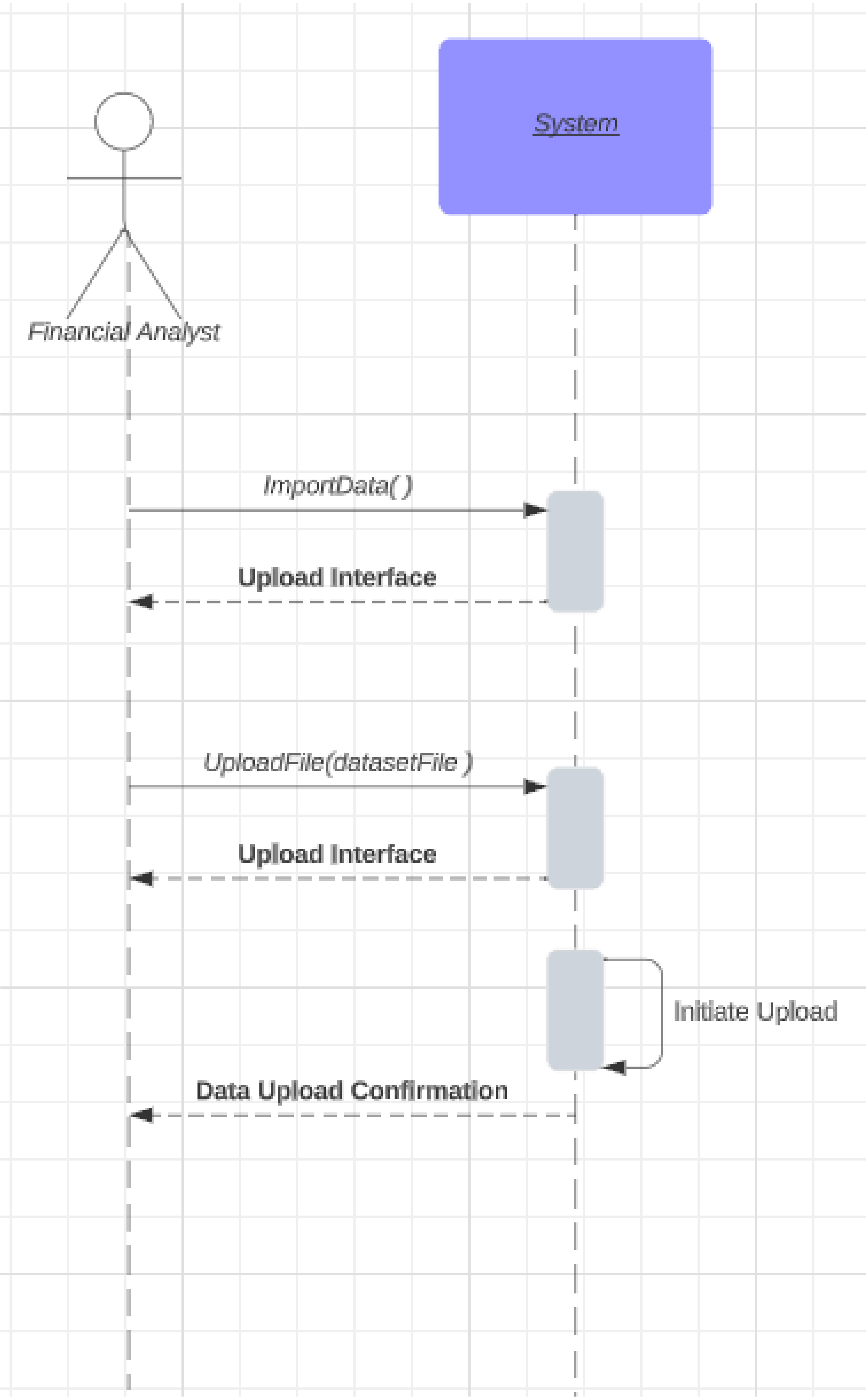


Figure 3.8: Upload Data

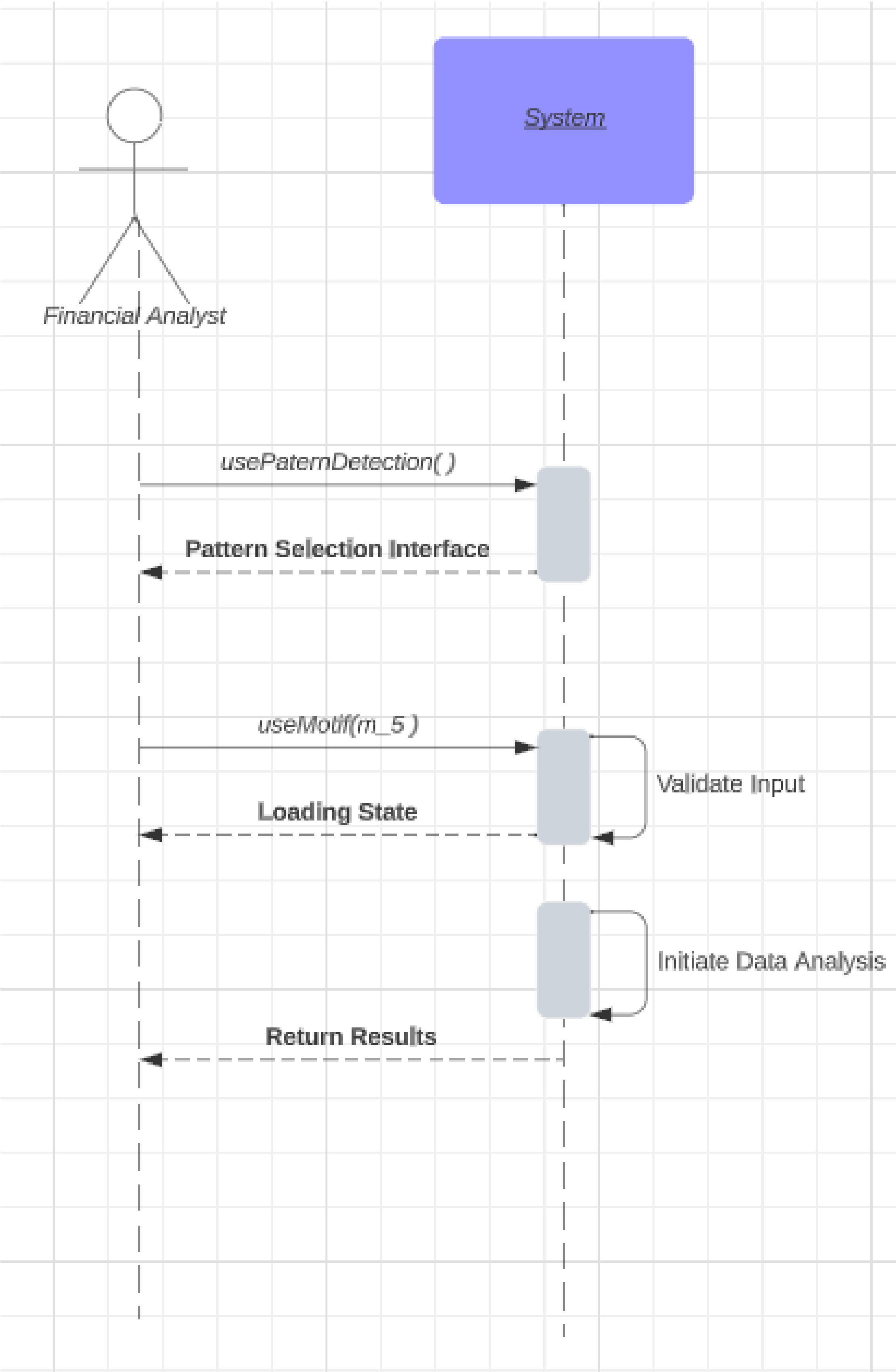


Figure 3.9: Motif Analysis

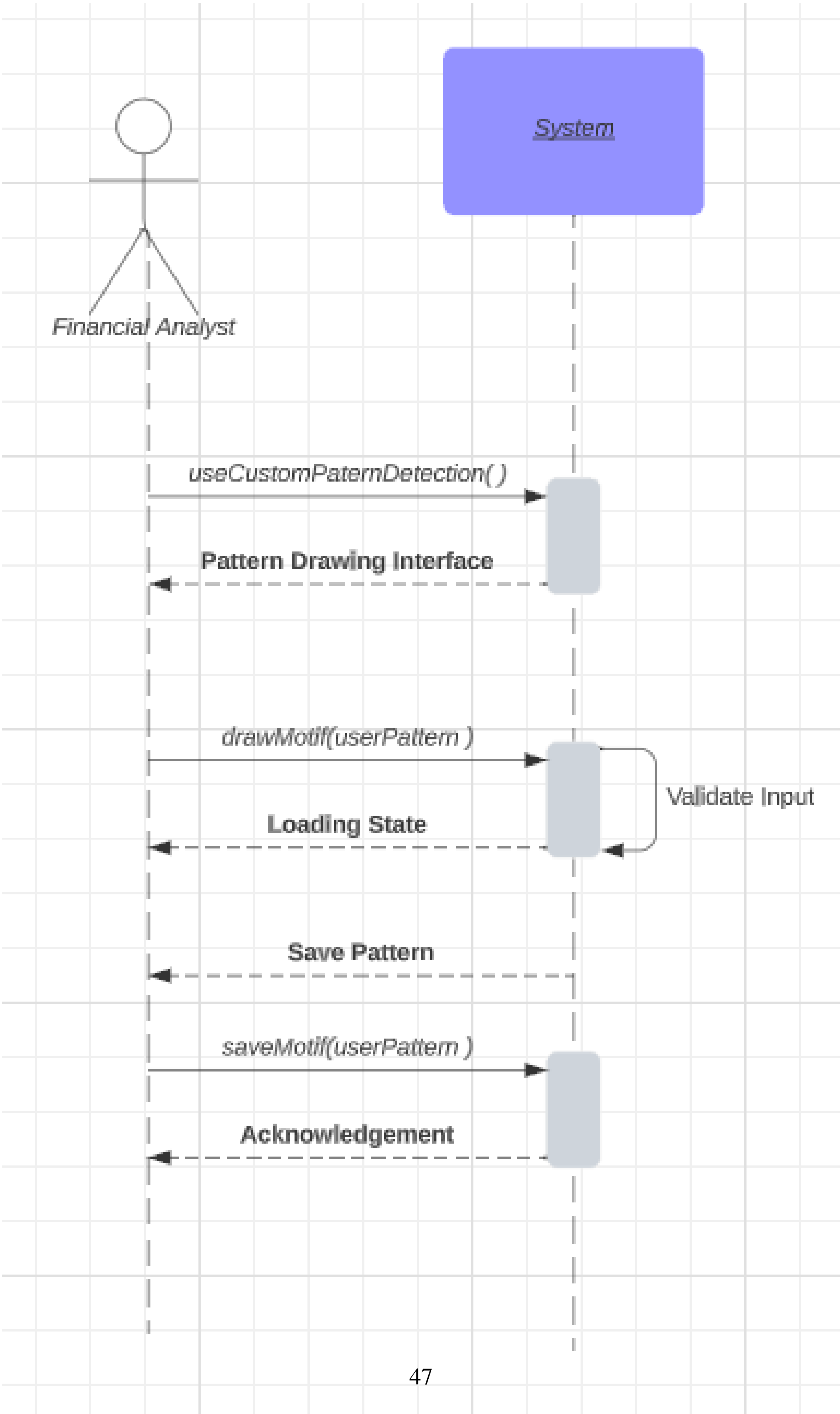


Figure 3.10: Draw Motif

3.3 Data Design

Our FYP doesn't necessitate the establishment of a database infrastructure since we won't be engaging in persistent data storage. Our primary focus revolves around directly utilizing a CSV dataset to detect financial fraud. As a result, an Entity-Relationship Diagram (ERD) isn't applicable in our context. Therefore, we'll be showcasing the JSON schema representing our dataset structure instead.

We used a Realistic Synthetic Financial dataset https://arxiv.org/abs/2306.16424 of money laundering for our FYP.

```
"$schema": "http://json-schema.org/draft-07/schema#",
"title": "H1-Large_Trans",
"type": "object",
"properties": {
  "Timestamp": {
    "type": "string",
    "format": "date-time",
    "description": "Timestamp of the transaction"
  },
  "FromBank": {
    "type": "integer",
    "description": "Bank ID of the sender"
  },
  "FromAccount": {
    "type": "string",
    "description": "Account number of the sender"
  },
  "ToBank": {
    "type": "integer",
    "description": "Bank ID of the receiver"
  },
  "ToAccount": {
    "type": "string",
    "description": "Account number of the receiver"
  },
  "AmountReceived": {
    "type": "number",
    "format": "float",
    "description": "Amount received in the transaction"
  },
```

```
  "ReceivingCurrency": {
    "type": "string",
    "description": "Currency in which the amount is received"
  },
  "AmountPaid": {
    "type": "number",
    "format": "float",
    "description": "Amount paid in the transaction"
  },
  "PaymentCurrency": {
    "type": "string",
    "description": "Currency used for payment"
  },
  "PaymentFormat": {
    "type": "string",
    "description": "Format of the payment (e.g., Reinvestment, Credit Card)"
  },
  "IsLaundering": {
    "type": "integer",
    "enum": [0, 1],
    "description": "Indicates if the transaction is associated with money launderin
  }
},
"required": [
  "Timestamp",
  "FromBank",
  "FromAccount",
  "ToBank",
  "ToAccount",
  "AmountReceived",
  "ReceivingCurrency",
  "AmountPaid",
  "PaymentCurrency",
  "PaymentFormat",
  "IsLaundering"
]
```

Chapter 4

Implementation and Testing

The implementation of NeuraSight involved utilizing various modules and libraries to achieve its functionality. The user interface was developed using Electron, while the Time Series Analysis module utilized Arema for analyzing and forecasting time series data. NetworkX was employed for implementing the Temporal Motifs module, providing functionality for detecting patterns in financial transaction data. Data preprocessing, exploration, and visualization tasks were handled using Pandas and NumPy libraries. For the Graph Neural Networks (GNNs) module, PyTorch was utilized, offering tools and algorithms for building and training neural networks tailored for fraud detection. Testing of NeuraSight included unit testing for each module to validate individual functionality, ensuring accurate detection of financial fraud, and integration testing to assess the interoperability of different modules within the system.

4.1 Algorithm Design

4.1.1 Generating Data Overview From a CSV File

```
function generateDataOverview(filePath):
    // Load the CSV file
    data = loadCSV(filePath)

    // Initialize overview structure
    overview = initializeOverviewStructure()

    // Iterate through each column in the dataset
    for each column in data.columns:
        // Update overview with basic statistics (e.g., unique values, missing valu
```

```
        updateBasicStats(overview, column, data[column])

        // Update overview with advanced statistics (e.g., mean, standard deviation
        updateAdvancedStats(overview, column, data[column])

    // Return the complete overview
    return overview
```

4.1.2 Running Time Series Analysis

```
function runTimeSeriesAnalysis(data, targetColumn):
    // Preprocess the data (e.g., handle missing values, normalize)
    preprocessedData = preprocessData(data)

    // Split the data into training and testing sets
    trainData, testData = splitData(preprocessedData, ratio=0.8)

    // Train the time series model on the training data
    model = trainTimeSeriesModel(trainData, targetColumn)

    // Use the model to make predictions on the test data
    predictions = makePredictions(model, testData)

    // Evaluate the model's performance
    evaluation = evaluateModel(predictions, testData[targetColumn])

    // Return the trained model and its evaluation metrics
    return model, evaluation
```

4.1.3 Finding Network Motifs within the Network Graphs

```
function findNetworkMotifs(graph, motifStructure):
    // Generate all possible subgraphs of the specified size from the main graph
    subgraphs = generateSubgraphs(graph, motifStructure.size)

    // Initialize a list to store identified motifs
    motifs = []

    // Iterate through each subgraph
```

```
    for each subgraph in subgraphs:
        // Check if the subgraph matches the motif structure
        if isIsomorphic(subgraph, motifStructure):
            // If it matches, add the subgraph to the motifs list
            motifs.append(subgraph)

    // Return the list of identified motifs
    return motifs
```

4.1.4 Running Graph Neural Networks to Detect Fraud

```
function runGNNDraudDetection(data, labels, semiSupervised=False):
    // Preprocess the data (e.g., normalization, handle missing values)
    preprocessedData = preprocessData(data)

    // Determine if the learning will be semi-supervised or fully supervised
    if semiSupervised:
        // Split the data into labelled and unlabelled datasets
        labelledData, unlabelledData = splitLabelledUnlabelled(preprocessedData, la

        // Train the semi-supervised GNN model
        model = trainSemiSupervisedGNN(labelledData, unlabelledData)
    else:
        // Train the supervised GNN model
        model = trainSupervisedGNN(preprocessedData, labels)

    // Return the trained GNN model
    return model
```

4.2 External APIs/SDKs

API and version	Description	Purpose of usage	API endpoint/function/class used
Electron (v13.1.7)	Framework for creating native applications with web technologies	Building the desktop application interface	BrowserWindow, ipcMain, ipcRenderer
PyTorch (v1.10.0)	Deep learning framework	Implementing and training Graph Neural Networks	torch.nn, torch.optim, torch.utils.data
NetworkX (v2.6.3)	Library for the creation, manipulation, and study of complex networks	Analyzing and visualizing the financial network	Graph, DiGraph, nx.motif.find$_{m}otifs$

4.3 Testing Details

We tested our modules of our project using Unit Testing methods.

4.3.1 Unit Testing

Unit testing is a level of software testing where individual units of a software/component are tested. Our goal was to verify that each module works perfectly on transactional data.

4.3.1.1 Unit Testing 1: Data Overview Module

Testing Objective: To verify the functionality of the Data Overview module in NeuraSight.

Verification Method:

1. Input various transactional datasets of different sizes and formats.
2. Ensure that the module provides accurate summary statistics such as total number of transactions, unique entities involved, and overall distribution of transaction amounts.
3. Confirm that the module displays relevant visualizations, such as histograms or pie charts, to represent the data overview.

4.3.1.2 Unit Testing 2: Time Series Analysis Module

Testing Objective: To ensure the accuracy and reliability of the Time Series Analysis module in NeuraSight.

Verification Method:

1. Input synthetic and real-world financial transaction datasets with known patterns of fraud.
2. Execute the Time Series Analysis module to detect anomalies and fraudulent activities.
3. Compare the module's output with expected results to validate its accuracy in identifying fraudulent transactions.
4. Assess the module's performance under various scenarios, including different time intervals and levels of noise in the data.

4.3.1.3 Unit Testing 3: Temporal Motifs Module

Testing Objective: To validate the functionality of the Temporal Motifs module in NeuraSight.

Verification Method:

1. Define custom motifs based on known patterns of fraudulent behavior in financial transactions.
2. Apply the Temporal Motifs module to transactional datasets containing these motifs.
3. Evaluate the module's ability to accurately identify instances of defined motifs within the dataset.
4. Test the module's flexibility by adjusting motif parameters and assessing its performance across various motif configurations.
5. Compare the module's results with expected outcomes to ensure its effectiveness in detecting temporal patterns indicative of fraud.

4.3.1.4 Unit Testing 4: Graph Neural Networks Module

Testing Objective: To verify the functionality of the Graph Neural Networks module in NeuraSight.

Verification Method:

1. Provide labeled or semi-labeled transactional datasets with known instances of fraud.
2. Train the Graph Neural Networks model using the provided datasets.
3. Validate the trained model's performance on test datasets by measuring metrics such as precision, recall, and F1 score.
4. Test the model's ability to generalize to new, unseen data by evaluating its performance on fresh transactional datasets.
5. Assess the module's scalability and efficiency in handling large-scale financial networks while maintaining detection accuracy.

4.4 Bibliography

- Popov, P. (2023). Success Stories: Applications and Benefits of Knowledge Graphs in Financial Services. *Journal Name*.
- Tian, Y. (2023). Transaction Fraud Detection via an Adaptive Graph Neural Network. *Journal Name*. Retrieved from https://arxiv.org/abs/2307.05633

- Amazon Web Services, Inc. Introduction to Amazon SageMaker. *Video*.
- Deloitte. Knowledge Graphs in Financial Services: Unlocking the Value.
- Liu, X., others. (2022). Graph Neural Networks for Financial Transaction Fraud Detection: A Featureless Approach.https://arxiv.org/abs/2205.13426
- Tribune. (2023). Rs70b Money Laundering Uncovered.
- Tribune. (2023). Massive Money Laundering Uncovered.
- Tribune. (2023). Rs13b Money Laundering Scam Unearthed.
- Altman, E., Egressy, B., Blanuša, J., Atasu, K. (2023). Realistic Synthetic Financial Transactions for Anti-Money Laundering Models.